Communication Lower Bounds for Distributed Convex Optimization: Partition Data on Features

Zihao Chen
Zhiyuan College
Shanghai Jiao Tong University
z.h.chen@sjtu.edu.cn

Luo Luo
Department of Computer Science and Engineering
Shanghai Jiao Tong University
ricky@sjtu.edu.cn

Zhihua Zhang
School of Mathematical Sciences
Peking University
zhzhang@math.pku.edu.cn

Abstract

Recently, there has been an increasing interest in designing distributed convex optimization algorithms under the setting where the data matrix is partitioned on features. Algorithms under this setting sometimes have many advantages over those under the setting where data is partitioned on samples, especially when the number of features is huge. Therefore, it is important to understand the inherent limitations of these optimization problems. In this paper, with certain restrictions on the communication allowed in the procedures, we develop tight lower bounds on communication rounds for a broad class of non-incremental algorithms under this setting. We also provide a lower bound on communication rounds for a class of (randomized) incremental algorithms.

1 Introduction

In this paper, we consider the following distributed convex optimization problem over m machines:

$$\min_{w \in \mathbb{R}^d} f(w; \theta).$$

Each machine knows the form of f but only has some partial information of θ. In particular, we mainly consider the case of *empirical risk minimization* (ERM) problems. Let $A \in \mathbb{R}^{n \times d}$ be a matrix containing n data samples with d features and $A_{j:}$ be the j-th row of the data matrix (corresponding to the j-th data sample). Then f has the form:

$$f(w) = \frac{1}{n} \sum_{j=1}^{n} \phi(w, A_{j:}), \tag{1}$$

where ϕ is some kind of convex loss.

In the past few years, many distributed optimizations algorithms have been proposed. Many of them are under the setting where data is partitioned on samples, i.e. each machine stores a subset of the data matrix A's rows (Zhang and Xiao 2015; Zhang, Wainwright, and Duchi 2012; Balcan et al. 2012; Boyd et al. 2011; Yang 2013; Lee, Ma, and Lin 2015; Jaggi et al. 2014; Ma et al. 2015). Meanwhile, as the dimension d can be enormously large, there has been an increasing interest in designing algorithms with the setting where the data is partitioned on features, i.e., each machine stores a subset of A's columns (Richtárik and Takáč 2013; Mareček, Richtárik, and Takáč 2015; Ma and Takáč 2016; Necoara and Clipici 2013; Lee and Roth 2015). Compared with algorithms under the sample partition setting, these algorithms have relatively less communication cost when d is large. In addition, as there is often no master machine in these algorithms, they tend to have more balanced workload on each machine, which is also an important factor affecting the performance of a distributed algorithm.

As communication is usually the bottleneck of distributed optimization algorithms, it is important to understand the fundamental limits of distributed optimization algorithms, i.e., how much communication an algorithm must need to reach an ϵ-approximation of the minimum value. Studying fundamental communication complexity of the distributed computing without any assumption is quite hard, letting alone continuous optimization. As for optimization, one alternative way is to derive lower bounds on communication rounds. The number of communication rounds is also an important metric as in many algorithms, faster machines often need to pause and wait for slower ones before they communicate, which can be a huge waste of time. Recently, putting some restrictions on communication allowed in each iteration, Arjevani and Shamir (2015) developed lower bounds on communication rounds for a class of distributed optimization algorithms under the sample partition setting, and these lower bounds can be matched by some existing algorithms. However, optimization problems under the feature partition setting are still not well understood.

Considering the increasing interest and importance of designing distributed optimization algorithms under the feature partition setting, in our paper, we develop tight lower bounds on communication rounds for a broad class of distributed optimization algorithms under this setting. Our results can provide deeper understanding and insights for designing optimization algorithms under this setting. To define the class of algorithms, we put some constraints on the form and amount of communication in each round, while keeping restrictions mild and applying to many distributed algorithms. We summarize our contributions as follows:

- For the class of smooth and λ-strongly convex functions with condition number κ, we develop a tight lower bound of $\Omega\left(\sqrt{\kappa}\log(\frac{\lambda\|w^* - w_0\|}{\epsilon})\right)$, which can be matched by a straightforward distributed version of accelerated gradient decent (Nesterov 2013) and also DISCO-F for quadratics (Ma and Takáč 2016), an variant of DISCO (Zhang and Xiao 2015) under the feature partition setting.

- For the class of smooth and (non-strongly) convex functions with Lipschitz smooth parameter L, we develop a tight lower bound of $\Omega\left(\sqrt{\frac{L}{\epsilon}}\|w^* - w_0\|\right)$, which is also matched by the distributed accelerated gradient decent.
- By slightly modifying the definitions of the algorithms, we define a class of incremental/stochastic algorithms under the feature partition setting and develop a lower bound of $\Omega\left((\sqrt{n\kappa}+n)\log(\frac{\|w^*-w_0\|\lambda}{\epsilon})\right)$ for λ-strongly convex functions with condition number κ.

Related Work The most revelant work should be (Arjevani and Shamir 2015), which studied lower bounds under the sample partition setting, and provided tight lower bounds on communication rounds for convex smooth optimization after putting some mild restrictions. More recently, Lee, Ma, and Lin (2015) provided a lower bound for another class of algorithms under that setting. Both these work as well as ours are based on some techniques used in non-distributed optimization lower bound analysis (Nesterov 2013; Lan 2015).

2 Notations and Preliminaries

We use $\|\cdot\|$ as Euclidean norm throughout the paper. For a vector $w \in \mathbb{R}^d$, we denote $w(i)$ as the i-th coordinate of the vector $w \in \mathbb{R}^d$. We let the coordinate index set $[d]$ be partitioned into m disjoint sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ with $\sum_{i=1}^m d_i = d$ and $\mathcal{S}_j = \{k \in [d] \big| \sum_{i<j} d_i < k \le \sum_{i\le j} d_i\}$ for $j = 1, 2, \ldots, m$. For a vector $w \in \mathbb{R}^d$, denote $w^{[j]}$ as a vector in $\mathbb{R}^{d_j}$ which equals to the segment of w on coordinates $\mathcal{S}_j$. For a set of vectors $\mathcal{V} \subseteq \mathbb{R}^d$, we define $\mathcal{V}^{[j]} = \{v^{[j]} \big| v \in \mathcal{V}\} \subseteq \mathbb{R}^{d_j}$. Then we define $f'_j(x) := \frac{\partial f(w)}{\partial w^{[j]}}\Big|_{w=x}$ and $f''_{ij}(x) := \frac{\partial^2 f(w)}{\partial w^{[i]} \partial w^{[j]}}\Big|_{w=x}$.

Through the whole paper, we use *partition-on-sample* and *partition-on-feature* to describe distributed algorithms or communication lower bounds under the settings where data is partitioned on samples and features respectively.

Then we list several preliminaries:

1. **Lipschitz continuity**. A function h is called Lipschitz continuous with constant L if
$$\forall x, y \in \text{dom } h \quad \|h(x) - h(y)\| \le L\|x - y\|.$$
2. **Lipschitz smooth and strongly convex**. A function h is called L-smooth and λ-strongly convex if
$$\frac{\lambda}{2}\|x-y\|^2 \le h(y) - h(x) - (x-y)^T \nabla h(y) \le \frac{L}{2}\|x-y\|^2$$
3. **Communication operations**. Here we list some common MapReduce types of communication operations (Dean and Ghemawat 2008) in an abstract level:
   (a) *One-to-all broadcast*. One-to-all broadcast is an operation that one machine sends identical data to all other machines.
   (b) *All-to-all broadcast*. All-to-all broadcast is an operation that each machines performs a one-to-all broadcast simultaneously.
   (c) *Reduce*. Consider the setting where each processor has p units of data, Reduce is an operation that combines the data items piece-wise (using some associative operator, such as addition or min), and make the result available at a target machine. For example, if each machine owns an $\mathbb{R}^d$ vector, then computing the average of the vectors needs a Reduce operation of an $\mathbb{R}^d$ vector.
   (d) *ReduceAll*. A ReduceAll operation can be viewed as a combination of a Reduce operation and a one-to-all broadcast operation.

3 Definitions and Framework

In this section we first describe a family of distributed optimization algorithms using m machines, and then modify it to get a family of incremental algorithms. At the beginning, the feature coordinates are partitioned into m sets and each machine owns the data columns corresponding to its coordinate set. We model the algorithms as iterative processes in multiple rounds and each round consists of a *computation phase* followed by a *communication phase*. For each machine we define a feasible set and during the computation phase, each machine can do some "cheap" communication and add some vectors to it. During the *communication phase* each machine can broadcast some limited number of points to all other machines. We also assume the communication operations are the common operations like broadcast, Reduce and ReduceAll.

3.1 Non-incremental Algorithm Family

Here we define the non-incremental algorithm class in a formal way:

Definition 1 (partition-on-feature distributed optimization algorithm family $\mathcal{F}^{\lambda,L}$). *We say an algorithm $\mathcal{A}$ for solving (1) with m machines belongs to the family $\mathcal{F}^{\lambda,L}$ of distributed optimization algorithms for minimizing L-smooth and λ-strongly convex functions ($\lambda = 0$ for non-strongly-convex functions) with the form (1) if the data is partitioned as follows:*

- *Let the coordinate index set $[d]$ be partitioned into m disjoint sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ with $\sum_{i=1}^m d_i = d$ and $\mathcal{S}_j = \{k \in [d] \big| \sum_{j<i} d_i < k \le \sum_{j\le i} d_i\}$ for $j = 1, 2, \ldots, m$. The data matrix $A \in \mathbb{R}^{n\times d}$ is partitioned column-wise as $A = [A_1, \ldots, A_m]$, where A_j consists of columns $i \in \mathcal{S}_j$. Each machine j stores A_j.*

and the machines do the following operations in each round:

1. *For each machine j, define a feasible set of vectors $\mathcal{W}_j \subseteq \mathbb{R}^{d_j}$ initialized to be $\mathcal{W}_j^{(0)} = \{0\}$. Denote $\mathcal{W}_j^{(k)}$ as machine j's feasible set $\mathcal{W}_j$ in the k-th round.*
2. ***Assumption on feasible sets.** In the k-th round, initially $\mathcal{W}_j^{(k)} = \mathcal{W}_j^{(k-1)}$. Then for a constant number of times,*

each machine j can add any w_j to $\mathcal{W}_j^{(k)}$ if w_j satisfies

$$w_j \in span\Big\{u_j,\ f_j'(u),\ (f_{jj}''(u)+D)v_j,\ f_{ji}''(u)v_i\ \Big|\ u^T=[{u_1}^T,\dots,{u_m}^T],\ u_j\in\mathcal{W}_j^{(k)},\ v_j\in\mathcal{W}_j^{(k)},\ u_i\in\mathcal{W}_i^{(k-1)},\ v_i\in\mathcal{W}_i^{(k-1)},\ i\neq j,\ D \text{ is diagonal}\Big\}. \tag{2}$$

3. ***Computation phase.*** *Machines do local computations and can perform no more than some constant times of Reduce/ReduceAll operations of an $\mathbb{R}^n$ vector or constant during the computation phase.*
4. ***Communication phase.*** *At the end of each round, each machine j can simultaneously broadcast no more than constant number of $\mathbb{R}^{d_j}$ vectors.*
5. *The final output after R rounds is $w^T=[w_1^T,\dots,w_m^T]$ satisfies $w_j\in\mathcal{W}_j^{(R)}$.*

We have several remarks on the defined class of algorithms:

- As described above, during the iterations to find w^*, machine j can only do updates on coordinates $\mathcal{S}_j$. This restriction is natural because machine j does not have much information about function f on other coordinates. Actually, almost all existing partition-on-feature algorithms satisfy this restriction (Richtárik and Takáč 2013; Mareček, Richtárik, and Takáč 2015; Ma and Takáč 2016; Necoara and Clipici 2013; Lee and Roth 2015).
- Similar to (Arjevani and Shamir 2015), we use $\mathcal{W}_j$ to define the restriction on the updates. This is not an explicit part of algorithms and machines do not necessarily need to store it, nor do they need to evaluate the points every time they add to the feasible sets. Although in most algorithms belonging to this family, each machine broadcasts the points it has added to the feasible set and store what other machines have broadcast in the communication phase (a simple example is the straightforward distributed implementation of gradient decent), we choose not to define feasible sets as physical sets that machines need to store to keep our results general.
- The assumption on the updates is mild and it applies to many partition-on-feature algorithms. It allows machines to perform preconditioning using local second order information or use partial gradient to update. It also allows to compute and utilize global second order information, since the span we define includes the $f_{ji}''(u)v_i$ term. Besides, we emphasize that putting such structural assumptions is necessary. Even if we could develop some assumption-free communication lower bounds, they might be too weak and have a large gap with upper bounds provided by existing algorithms, thus becoming less meaningful and cannot provide deeper understanding or insights for designing algorithms.
- During the computation phase, we assume each local machines can perform unbounded amount of computation and only limited amount of communication. Note that this part of communication is a must in many partition-on-feature algorithms, usually due to the need of computing partial gradients $f_j'(w)$. However, for some common loss functions ϕ, such as (regularized) squared loss, logistic loss and squared hinge loss, computing $f_j'(w)$ for all $j\in[m]$ in total only needs a ReduceAll operation of an $\mathbb{R}^n$ vector (Richtárik and Takáč 2013). In some gradient (or partial gradient) based algorithms (Richtárik and Takáč 2013; Mareček, Richtárik, and Takáč 2015), communication to compute partial gradients are the only need of communication in the computation phase. Besides, some algorithms like DISCO-F (Ma and Takáč 2016) need to compute $(\nabla f(w)u)^{[j]}$. For loss functions like squared loss, logistic loss and squared hinge loss, it only requires the same amount of communication as computing partial gradients, i.e. a ReduceAll operation of an $\mathbb{R}^n$ vector (Ma and Takáč 2016).
- Here we summarize the total communication allowed in each round. We use $\tilde{\mathcal{O}}$ to denote asymptotic bounds hiding constants and factors logarithmic in the required accuracy of the solution. During the computation phase, each machine can do constant times of ReduceAll operations of $\tilde{\mathcal{O}}(n)$ bits. During the communication phase, each machine j can broadcast $\tilde{\mathcal{O}}(d_j)$ bits, this can be viewed as performing constant times of ReduceAll operation of an $\mathbb{R}^d$ vector. Therefore, the total communication allowed in each round is no more than ReduceAll operations of $\tilde{\mathcal{O}}(n+d)$ bits. The amount of communication allowed in our partition-on-feature algorithm class is relatively small, compared with the partition-on-sample algorithm class described in (Arjevani and Shamir 2015), which allows $\tilde{\mathcal{O}}(md)$ bits of one-to-all broadcast in each round. This is due to the partition-on-feature algorithms' advantage on communication cost.
- We also emphasize that the communication allowed in the entire round is moderate. On one hand, if we only allow too little communication then the information exchange between machines is not enough to perform efficient optimization. As a result, hardly no practical distributed algorithm could satisfy the requirement, which will diminish the generality of our results. On the other hand, if we allow too much communication in each round, our assumption on the feasible sets can be too strong. For example if we allow enough communication for all machines to broadcast their entire local data, then the machines only need one communication round to find out a solution up to any accuracy ϵ.
- Our assumption that $\mathcal{W}_j$ is initialized as $\{0\}$ is merely for convenience and we just need to shift the function via $\bar{f}(w)=f(w+w_0)$ for another starting point w_0.

3.2 Incremental Algorithm Family

To define the class of incremental/stochastic algorithms $\mathcal{I}^{\lambda,L}$ under the feature partition setting, we slightly modify the definition of $\mathcal{F}^{\lambda,L}$ by replacing assumption on feasible set (2) with the following while keeping the rest unchanged:

Assumption on feasible sets for $\mathcal{I}^{\lambda,L}$. In the k-th round, initially $\mathcal{W}_j^{(k)} = \mathcal{W}_j^{(k-1)}$. Next for a constant number of times, each machine j chooses $g(w) := \phi(w, A_{l:})$ for some (possibly random) l and adds any w_j to $\mathcal{W}_j^{(k)}$ if w_j satisfies

$$\begin{aligned} w_j \in \operatorname{span}\Big\{ u_j,\ g_j'(u),\ (g_{jj}''(u) + D)v_j,\ g_{ji}''(u)v_i \ \Big| \\ u^T = [{u_1}^T, \dots, {u_m}^T],\ u_j \in \mathcal{W}_j^{(k)},\ v_j \in \mathcal{W}_j^{(k)}, \\ u_i \in \mathcal{W}_i^{(k-1)},\ v_i \in \mathcal{W}_i^{(k-1)},\ i \neq j,\ D \text{ is diagonal} \Big\}. \quad (3) \end{aligned}$$

4 Main Results

In this section, we present our main theorems followed by some discussions on their implications.

First, we present a lower bound on communication rounds for algorithms in $\mathcal{F}^{\lambda,L}$:

Theorem 2. *For any number m of machines, any constants $\lambda, L, \epsilon > 0$, and any distributed optimization algorithm $\mathcal{A} \in \mathcal{F}^{\lambda,L}$, there exists a λ-strongly convex and L-smooth function $f(w)$ with condition number $\kappa := \frac{L}{\lambda}$ over $\mathbb{R}^d$ such that if $w^* = \arg\min_{w \in \mathbb{R}^d} f(w)$, then the number of communication rounds to obtain $\hat{w}$ satisfying $f(\hat{w}) - f(w^*) \leq \epsilon$ is at least*

$$\Omega\left(\sqrt{\kappa}\log\left(\frac{\|w^*\|\lambda}{\epsilon}\right)\right) \quad (4)$$

for sufficiently large d.

Similarly, we have a lower bound for algorithms in $\mathcal{F}^{0,L}$ for minimizing smooth convex functions:

Theorem 3. *For any number m of machines, any constants $L > 0, \epsilon > 0$, and any distributed optimization algorithm $\mathcal{A} \in \mathcal{F}^{0,L}$, there exists a L-smooth convex function $f(w)$ over $\mathbb{R}^d$ such that if $w^* = \arg\min_{w \in \mathbb{R}^d} f(w)$, then the number of communication rounds to obtain $\hat{w}$ satisfying $f(\hat{w}) - f(w^*) \leq \epsilon$ is at least*

$$\Omega\left(\sqrt{\frac{L}{\epsilon}}\|w^*\|\right) \quad (5)$$

for sufficiently large d.

Then we contrast our lower bound for smooth strongly convex functions with some existing algorithms and the upper bounds provided by their convergence rate. The comparisons indicate that our lower bounds are tight. For some common loss functions, this can be matched by a straightforward distributed implementation of accelerated gradient decent (Nesterov 2013) and it is easy to verify that it satisfies our definition: let all machines compute their own partial gradients and aggregate to form a gradient. This straightforward distributed version of accelerated gradient decent achieves a round complexity of $O\left(\sqrt{\kappa}\log(\frac{\|w^*-w_0\|\lambda}{\epsilon})\right)$, which matches our lower bound exactly. Similarly, our lower bound on smooth (non-strongly) convex functions can also be matched by the distributed accelerated gradient decent.

Recall that our definition of the algorithm class includes some types of distributed second order algorithms, for example DISCO-F (Ma and Takáč 2016). The number of communication rounds DISCO-F needs to minimize general quadratic functions is $\mathcal{O}\left(\sqrt{\kappa}\log(\frac{\|w^*-w_0\|}{\epsilon})\right)$, which also matches our bound with respect to κ and ϵ. This indicates that distributed second order algorithms may not achieve a faster convergence rate than first order ones if only linear communication is allowed.

Next we present a lower bound on communication rounds for algorithms in $\mathcal{I}^{\lambda,L}$:

Theorem 4. *For any number m of machines, any constants $\lambda, L, \epsilon > 0$, and any distributed optimization algorithm $\mathcal{A} \in \mathcal{I}^{\lambda,L}$, there exists a λ-strongly convex and L-smooth function $f(w)$ with condition number $\kappa := \frac{L}{\lambda}$ over $\mathbb{R}^d$ such that if $w^* = \arg\min_{w \in \mathbb{R}^d} f(w)$, then the number of communication rounds to obtain $\hat{w}$ satisfying $\mathbb{E}\left[f(\hat{w}) - f(w^*)\right] \leq \epsilon$ is at least*

$$\Omega\left(\left(\sqrt{n\kappa} + n\right)\log\left(\frac{\|w^*\|\lambda}{\epsilon}\right)\right) \quad (6)$$

for sufficiently large d.

5 Proof of Main Results

In this section, we provide proofs of Theorem 2 and Theorem 4. The proof framework of these theorems are based on (Nesterov 2013). As Theorem 3 can be obtained by replacing (Nesterov 2013, Lemma 2.1.3) with Corollary 6 (see below) in the proof of (Nesterov 2013, Theorem 2.1.6), we will not discuss it here for simplicity.

5.1 Proof of Theorem 2

The idea is to construct a "hard" function so that all algorithms in the class we defined could not optimize well in a small number of rounds: in each round only one of the machines can do a constant steps of "progress" while other machines stay "trapped" (see Lemma 5).

Without loss of generality, we assume that in every round, each machine only add one vector to its feasible set and the bound does not change asymptotically.

First, we construct the following function like (Lan 2015)

$$f(w) = \frac{\lambda(\kappa-1)}{4}\left[\frac{1}{2}w^T A w - \langle e_1, w\rangle\right] + \frac{\lambda}{2}\|w\|^2, \quad (7)$$

where A is a tridiagonal matrix in $\mathbb{R}^{d\times d}$ with the form

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & \frac{\sqrt{\kappa}+3}{\sqrt{\kappa}+1} \end{bmatrix}.$$

It is easy to verify that $f(w)$ is λ-strongly convex with condition number κ.

After K rounds of iteration, let $\mathcal{E}_{t,d} := \{x \in \mathbb{R}^d \big| x(i) = 0,\ t+1 \leq i \leq d\}$ and $\mathcal{W}^{(K)} := \{[w_1^T, \dots, w_m^T]^T \big| w_j \in \mathcal{W}_j^{(K)}, j = 1, \dots, m\}$.

Then we have the following lemma:

Lemma 5. *If $\mathcal{W}^{(K)} \subseteq \mathcal{E}_{K,d}$ for some $K \le d-1$, then we have $\mathcal{W}^{(K+1)} \subseteq \mathcal{E}_{K+1,d}$.*

Proof. First we recall the assumption on $\mathcal{W}_j$'s in (2):

$$w_j \in \text{span}\Big\{u_j,\ f_j'(u),\ (f_{jj}''(u)+D)v_j,\ f_{ji}''(u)v_i\ \Big|\ u^T = [{u_1}^T, \dots, {u_m}^T],\ u_j \in \mathcal{W}_j^{(k)},\ v_j \in \mathcal{W}_j^{(k)},\ u_i \in \mathcal{W}_i^{(k-1)},\ v_i \in \mathcal{W}_i^{(k-1)},\ i \ne j,\ D \text{ is diagonal}\Big\}.$$

We just need to prove that for any vector $u,\ v \in \mathcal{W}^{(K)} \subseteq \mathcal{E}_{K,d}$,

$$u_j, f_j'(u),\ (f_{jj}''(u)+D)v_j,\ f_{ji}''(u)v_i \in \mathcal{E}_{K+1,d}^{[j]}$$

For convenience of the proof, we partition A as follow

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}, \tag{8}$$

where $A_{11} \in \mathbb{R}^{a\times a}$, $A_{22} \in \mathbb{R}^{b\times b}$, $A_{33} \in \mathbb{R}^{c\times c}$ and $a = \sum_{i<j} d_i$, $b = d_j$, $c = \sum_{i>j} d_i$. Let $x_1 = [u_1^T, u_2^T, \dots, u_{j-1}^T]^T$, $x_2 = u_j$, and $x_3 = [u_{j+1}^T, \dots, u_m^T]^T$. Then we obtain

$$f_j'(u) = \begin{cases} \Big[\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I\Big]x_2 + \\ \frac{\lambda(\kappa-1)}{2}(A_{21}x_1 + A_{23}x_3) & j \ne 1 \\ \Big[\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I\Big]x_2 + \\ \frac{\lambda(\kappa-1)}{2}(A_{21}x_1 + A_{23}x_3) - \frac{\lambda(\kappa-1)}{4}e_1^{[1]} & j = 1 \end{cases}$$

and

$$f_{jj}''(u) = \frac{\lambda(\kappa-1)}{4}A_{22} + \lambda I.$$

As $f_{jj}''(u) + D$ is a tridiagonal matrix, we have

$$(f_{jj}''(u)+D)v_j \in \mathcal{E}_{K+1,d}^{[j]}. \tag{9}$$

Using the fact that

$$A_{21} = \begin{bmatrix} 0 & \cdots & 0 & -1 \\ 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} A_{23} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \\ -1 & 0 & \cdots & 0 \end{bmatrix},$$

we can rewrite f_j' as

$$f_j'(u) = \begin{cases} \Big[\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I\Big]x_2 + \\ \frac{\lambda(\kappa-1)}{2}[x_1(a), 0, \dots, 0, x_3(1)]^T, & j \ne 1 \\ \Big[\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I\Big]x_2 + \\ \frac{\lambda(\kappa-1)}{2}[\frac{1}{2}, 0, \dots, 0, x_3(1)]^T. & j = 1 \end{cases}$$

As $\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I$ is a tridiagonal matrix, then

$$\Big[\frac{\lambda(\kappa-1)}{2}A_{22} + \lambda I\Big]x_2 \in \mathcal{E}_{K+1,d}^{[j]}.$$

We can also obtain the following by some simple discussions on different cases:

$$[\frac{1}{2}, 0, \dots, 0, x_3(1)]^T,\ [x_1(a), 0, \dots, 0, x_3(1)]^T \in \mathcal{E}_{K+1,d}^{[j]}$$

Therefore, we conclude that

$$f_j'(w) \in \mathcal{E}_{K+1,d}^{[j]}. \tag{10}$$

Besides, note that

$$f_{ji}''(u)v_i = \begin{cases} [0, \dots, 0, -v_i(1)]^T & i = j+1 \\ [-v_i(d_i), 0, \dots, 0]^T & i = j-1 \\ [0, 0, \dots, 0]^T & \text{otherwise} \end{cases}$$

Similarly, using some simple discussions on different cases we obtain

$$f_{ji}''(u)v_i \in \mathcal{E}_{K+1,d}^{[j]} \tag{11}$$

Hence when $\mathcal{W}^{(K)} \subseteq \mathcal{E}_{K,d}$, combining (9) (10) and (11) we have for all $u \in \mathcal{W}^T$ and j

$$u_j,\ f_j'(u),\ (f_{jj}''(u)+D)v_j,\ f_{ji}''(u)v_i \in \mathcal{E}_{K+1,d}^{[j]}.$$

which implies the newly added w_j satisfies

$$w_j \in \mathcal{E}_{K+1,d}^{[j]}$$

Therefore, we prove that $\mathcal{W}^{(K+1)} \subseteq \mathcal{E}_{K+1,d}$. □

Applying Lemma 5 recursively we can get the following corollary:

Corollary 6. *After $K \le d$ rounds, we have $\mathcal{W}^{(K)} \subseteq \mathcal{E}_{K,d}$.*

With Corollary 6, we now proceed to finish the proof of Theorem 2. First, we can find w^* by the first order optimality condition

$$f'(w^*) = \Big(\frac{\lambda(\kappa-1)}{4}A + \lambda I\Big)w^* - \frac{\lambda(\kappa-1)}{4} = 0,$$

which implies

$$\Big(A + \frac{4}{\kappa-1}I\Big)w^* = e_1.$$

The coordinate form of above equation is

$$\begin{aligned} 2\frac{\kappa+1}{\kappa-1}w^*(1) - w^*(2) &= 1, \\ w^*(k+1) - 2\frac{\kappa+1}{\kappa-1}w^*(k) + w^*(k-1) &= 0, \\ 2 \le k \le d-2, \\ -x^*(d-1) + \left(\frac{4}{\kappa-1} + \frac{\sqrt{\kappa}+3}{\sqrt{\kappa}+1}\right)x^*(d) &= 0, \end{aligned}$$

and let q be the smallest root of the following equation

$$q^2 - \frac{2\kappa+2}{\kappa-1}q + 1 = 0,$$

that is $q = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$. Then w^* satisfies $w^*(i) = q^i$ for $1 \le i \le d$. Hence,

$$\|w^*\|^2 = \sum_{i=1}^d [w^*(i)]^2 = \sum_{i=1}^d q^{2i} = \frac{q^2(1-q^{2d})}{1-q^2}$$

Let $w^{(k)}$ be any point in $\mathcal{W}^{(k)}$ after k rounds iterations ($k \le d$), applying Corollary 6 we have

$$\begin{aligned}
\|w^{(k)} - w^*\|^2 &\ge \sum_{i=k+1}^{d} [w^*(i)]^2 = \sum_{i=k+1}^{d} q^{2i} \\
&= \frac{q^{2(k+1)}[1 - q^{2(d-k+2)}]}{1 - q^2} \\
&= \frac{1 - q^{2(d-k+2)}}{1 - q^{2d}} q^{2k} \|w^*\|^2 \\
&\ge \frac{1 - q^4}{1 - q^{2d}} q^{2k} \|w^*\|^2 \\
&\ge \frac{1 - q}{1} q^{2k} \|w^*\|^2
\end{aligned}$$

Combing the above inequality and the optimal condition of strongly-convex function, we obtain

$$\begin{aligned}
f(w^{(k)}) - f(w^*) &\ge \frac{\lambda}{2} \|w^{(k)} - w^*\|^2 \\
&\ge \frac{\lambda(1-q)}{2} q^{2k} \|w^*\|^2 \\
&= \frac{\lambda}{2} \frac{2}{\sqrt{\kappa}+1} \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2k} \|w^*\|^2 \\
&\ge \frac{\lambda}{\sqrt{\kappa}+1} \exp\left(-\frac{4k}{\sqrt{\kappa}+1}\right) \|w^*\|^2.
\end{aligned}$$

Thus if $f(w^{(k)}) - f(w^*) \le \epsilon$, then we have

$$\begin{aligned}
k &\ge \frac{\sqrt{\kappa}-1}{4} \log\left(\frac{\lambda \|w^*\|^2}{(\sqrt{\kappa}+1)\epsilon}\right) \\
&= \Omega\left(\sqrt{\kappa} \log\left(\frac{\lambda \|w^*\|}{\epsilon}\right)\right),
\end{aligned}$$

which completes the proof.

5.2 Proof of Theorem 4

With a slight abuse of notation, we construct the following separable strongly convex function:

$$f(w) := \frac{1}{m} \sum_{j=1}^{m} \phi_j(w_j), \tag{12}$$

where $w^T = [w_1^T, \dots, w_m^T]$ and $\phi_j(w_j)$ is also a separable strongly convex function with form of

$$\begin{aligned}
\phi_j(w_j) = \sum_{i=1}^{\frac{n}{m}} \Bigg[&\frac{\lambda(\kappa-1)}{4} \left(\frac{1}{2} w_{j,i}^T A_{j,i} w_{j,i} - \langle e_1, w_{j,i} \rangle\right) \\
&+ \frac{\lambda}{2} \|w_{j,i}\|^2 \Bigg],
\end{aligned} \tag{13}$$

where $w_j^T = [w_{j,1}^T, \dots, w_{j,n}^T]$ and $A_{j,i}$ is a tridiagonal matrix in $\mathbb{R}^{d_j \times d_j}$ given by

$$A_{j,i} = \begin{bmatrix}
2 & -1 & 0 & \cdots & 0 & 0 & 0 \\
-1 & 2 & -1 & \cdots & 0 & 0 & 0 \\
0 & -1 & 2 & \cdots & 0 & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & \cdots & -1 & 2 & -1 \\
0 & 0 & 0 & \cdots & 0 & -1 & \frac{\sqrt{\kappa}+3}{\sqrt{\kappa}+1}
\end{bmatrix}.$$

It is simple to veriy that f is a λ-strongly convex function with condition number κ. Note that f is a quadratic function with its coefficient matrix being a block diagonal matrix, in which each block matrix is tridiagonal. As f is separable, the setting here can be viewed as each machine j has all information (all data samples in all coordinates) of f's component ϕ_j, and all machines simultaneously do non-distributed incremental opitmizaion over their local data. To get the lower bound, note that in "clever" algorithms, machine j only chooses data samples among the ones corresponding to ϕ_j in each round, then we have

$$\begin{aligned}
&\mathbb{E}[\|w^{(k)} - w^*\|^2] \\
\ge\ &\mathbb{E}\left[\sum_{j=1}^{m} \|w_j^{(k)} - w_j^*\|^2\right] \\
\ge\ &\sum_{j=1}^{m} \left[\frac{1}{2} \exp\left(-\frac{4k\sqrt{\kappa}}{n(\sqrt{\kappa}+1)^2 - 4\sqrt{\kappa}}\right) \|w_j^*\|^2\right] \\
=\ &\frac{1}{2} \exp\left(-\frac{4k\sqrt{\kappa}}{n(\sqrt{\kappa}+1)^2 - 4\sqrt{\kappa}}\right) \|w^*\|^2.
\end{aligned}$$

The second inequality is according to (Lan 2015, Theorem 3) and Corollary 6.

Finally by (Lan 2015, Corollary 3), we get if $\mathbb{E}\left[f(w^{(k)}) - f(w^*)\right] \le \epsilon$, then

$$k \ge \Omega\left(\left(\sqrt{n\kappa} + n\right) \log\left(\frac{\|w^*\|\lambda}{\epsilon}\right)\right), \tag{14}$$

for sufficiently large d.

6 Conclusion

In this paper we have defined two classes of distributed optimization algorithms under the setting where data is partitioned on features: one is a family of non-incremental algorithms and the other is incremental. We have presented tight lower bounds on communication rounds for non-incremental class of algorithms. We have also provided one lower bound for incremental class of algorithms but whether it is tight remains open.

The tightness informs that one should break our definition when trying to design optimization algorithms with less communication rounds than existing algorithms. We also emphasize that our lower bounds are important as they can provide deeper understanding on the limits of some ideas or techniques used in distributed optimization algorithms, which may provide some insights for designing better algorithms.

To the best of our knowledge, this is the first work to study communication lower bounds for distributed optimization algorithms under the setting where data is partitioned on features.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 61572017).

References

Arjevani, Y., and Shamir, O. 2015. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems*, 1747–1755.

Balcan, M.-F.; Blum, A.; Fine, S.; and Mansour, Y. 2012. Distributed learning, communication complexity and privacy. *arXiv preprint arXiv:1204.3514*.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3(1):1–122.

Dean, J., and Ghemawat, S. 2008. Mapreduce: simplified data processing on large clusters. *Communications of the ACM* 51(1):107–113.

Jaggi, M.; Smith, V.; Takác, M.; Terhorst, J.; Krishnan, S.; Hofmann, T.; and Jordan, M. I. 2014. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, 3068–3076.

Lan, G. 2015. An optimal randomized incremental gradient method. *arXiv preprint arXiv:1507.02000*.

Lee, C.-p., and Roth, D. 2015. Distributed box-constrained quadratic optimization for dual linear svm. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 987–996.

Lee, J.; Ma, T.; and Lin, Q. 2015. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*.

Ma, C., and Takáč, M. 2016. Distributed inexact damped newton method: Data partitioning and load-balancing. *arXiv preprint arXiv:1603.05191*.

Ma, C.; Smith, V.; Jaggi, M.; Jordan, M. I.; Richtárik, P.; and Takáč, M. 2015. Adding vs. averaging in distributed primal-dual optimization. *arXiv preprint arXiv:1502.03508*.

Mareček, J.; Richtárik, P.; and Takáč, M. 2015. Distributed block coordinate descent for minimizing partially separable functions. In *Numerical Analysis and Optimization*. Springer. 261–288.

Necoara, I., and Clipici, D. 2013. Parallel coordinate descent methods for composite minimization: convergence analysis and error bounds.

Nesterov, Y. 2013. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Richtárik, P., and Takáč, M. 2013. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059*.

Yang, T. 2013. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, 629–637.

Zhang, Y., and Xiao, L. 2015. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*.

Zhang, Y.; Wainwright, M. J.; and Duchi, J. C. 2012. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, 1502–1510.